

VISIT...

LANZAROTE
Caliente.COM

Comparison of Lifting and B-spline DWT Implementations for Implantable Neuroprosthetics.

Awais M. Kamboh, and Andrew Mason
Michigan State University, East Lansing, MI, USA
{kambohaw, mason}@msu.edu

Abstract— The discrete wavelet transform (DWT) has been shown to provide exceptionally efficient data compression for neural records. This paper describes area-power minimized hardware implementations of the lifting and B-spline DWT schemes for multi-level, multi-channel DWT and analyzes their performance tradeoffs for implantable neuroprosthetics. The lifting scheme is shown to be increasingly superior for a larger number of input channels.

I. INTRODUCTION AND MOTIVATION

Neuroprosthetic devices that utilize microelectrode arrays (MEAs) to monitor ensemble neural activity show great promise in many biomedical applications. However, the ability to implant devices into living bodies is limited by many challenges. Among these, two severe challenges are low area consumption and low power dissipation. Large area results in physical implantation challenges whereas high power dissipation could be very damaging to the tissue and nerves neighboring the implanted device.

For chronically implanted MEA devices, the most attractive means of data retrieval is wireless transmission, which eliminates the need for hardwired connections. However, the tremendous magnitude of raw data would require a prohibitively large transmission bandwidth. Likewise, extracting all of the useful information from MEA signals requires processing too complex to be implemented within an implanted system. These limitations can be overcome by data compression before transmission, which can be effectively achieved by performing a discrete wavelet transform (DWT) of the sampled neural data [1]. The resulting transform coefficients give a sparse representation of the signal. The non-zero DWT coefficients can then be encoded using a lossless encoding scheme and transmitted to the extra-cranial or extra-cutaneous processing units.

Modern neuroprosthetic systems must permit multiple signal channels to be sampled simultaneously. Furthermore, a DWT with multiple decomposition levels provides more accurate information about the original signal than a single-level DWT. Thus, to achieve the compression required for wireless neuroprosthetic implants, area and power efficient hardware is needed that can perform multi-channel, multi-level DWT in real time. Contrary to most modern

applications of DWT, neuroprosthetics can afford long computation intervals, up to 40 μ sec [1], relaxing hardware requirements.

Integer lifting and B-spline DWT factorization schemes have very efficient hardware implementations. The lifting approach reduces the required arithmetic operations and requires less memory at the expense of a longer critical path [2, 3]. The B-spline factorization reduces the critical path delay by converting some of the required multiply operations into less computationally intensive shift-and-add operations [3]. Recent efforts to optimize DWT hardware have concentrated on increasing throughput at the expense of area and power [2, 3]. In contrast, this paper identifies and compares optimal implementations of the lifting and B-spline architectures where chip area and power have priority.

II. THEORETICAL FRAMEWORK

Mallat's algorithm [4] has been used traditionally for evaluating the wavelet transform. It involves recursively convolving the signal through two decomposition filters and downsampling the result to obtain the approximation and detail coefficients at every decomposition level. Because the 'symmlet' family of wavelet basis has been shown to provide a near optimal compression of neural signals [1], this paper will focus on symmlet basis of order 4.

A. Lifting Factorization

The lifting scheme can be used to express any FIR wavelet transform by a sequence of "predict" and "update" lifting steps, S_n and T_n , respectively, as can be seen in Fig. 1. Mathematically, a lifting DWT is implemented by splitting the data into even and odd samples and applying the S_n and T_n filters simultaneously. The last step is a multiplication by

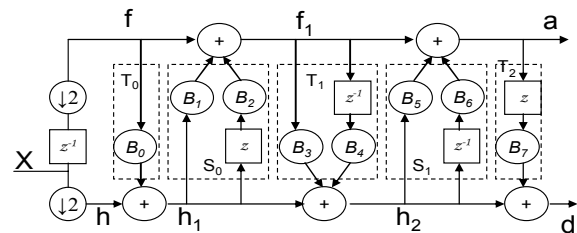


Figure 1. Lifting scheme for DWT.

a scaling factors K and $1/K$, which can be taken care of at the decoding side and is thus ignored in the following discussion.

B. B-spline Factorization

Wavelet filters can be decomposed as

$$\begin{aligned} H(z) &= (1+z)^M Q(z) \cdot h_o \\ L(z) &= (1-z)^M R(z) \cdot l_o \end{aligned} \quad (1)$$

where $(1 \pm z)^M$ are called the B-spline factors and the remaining factors are the distributed filters [3]. Expanding all multiplications in the B-spline factors allows them to be replaced by less computationally demanding shift and add operations. For symmlet 4 filters, the best decomposition results for $M=4$, where $(1 \pm z)^M$ can be represented by $(1+6z^2+z^4) \pm (4z^{-1}+4z^{-3})$. Fig. 2 describes the B-spline DWT implementation using symmlet 4 filters. The left-hand portion splits the input stream and implements B-spline factors of (1) using shifters and adders, as shown in a blowup box in Fig. 2. The right-hand portion implements the

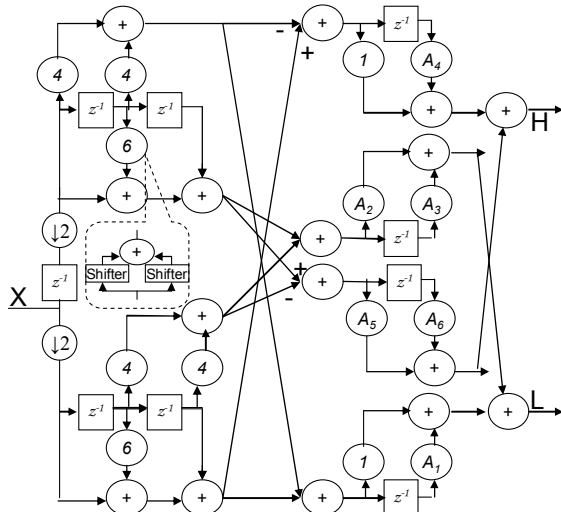


Figure 2. B-spline architecture for symmlet4 based DWT.

distributed factors, where a hardware multiplier is required.

C. Algorithmic Comparison

Lifting does not preserve causality, and two of its filters (S_0 and T_2 in Fig. 1) introduce output latency. On the other hand, B-spline does not have any causality issues, and incurs only a computational delay. However, because of the low biologically relevant sampling rate in neuroprosthetic applications (~ 25 kHz [1]), latency is not a critical limit to real-time operation.

To calculate a DWT using symmlet 4, the standard convolution based filter requires 14 multiplication and 14 additions. The lifting scheme reduces computations to only eight multiplications and eight additions. For B-spline factorization, the total multiplications and additions required

are reduced to 12 and 16, respectively. However, six of the multiplications can be implemented using shift-and-add operations, resulting in a minimized B-spline implementation with only six multiplications and 18 additions. Therefore, when multiplications have a much higher penalty than additions, which is particularly true for floating point operations, B-spline has the advantage.

III. HARDWARE IMPLEMENTATION

A. Lifting Implementation

The symmlet4 lifting filters of can be described by

$$\begin{aligned} P_{-1} &= h + B_0 f \\ Q_{-1} &= f + B_1 P_{-1} + B_2 P_0 \\ R_0 &= P_0 + B_3 Q_0 + B_4 Q_{-1} \\ a_0 &= Q_0 + B_5 R_0 + B_6 R_{-1} \\ d_{-1} &= R_{-1} + B_7 a_0 \end{aligned} \quad (2)$$

where each equation represents one filter step, a and d are the approximation and detail results, P , Q , and R are intermediate results, and B_i are the eight constant filter coefficients [5]. The variable subscripts in (2) represent the time sample, i.e., 0 represents the current sample, -1 represents the previous sample, and so on.

The lifting equations in (2) show a noticeable regularity in the required computations; all arithmetic operations can be expressed in the general form of $W = X + B_i Y + B_j Z$ (for the first and last steps Y or Z is set to zero). This regularity can be exploited to minimize hardware by implementing an optimized lifting computation core (CC_L) with two multiplications and two additions, as shown in Fig. 3(a). A single hardware block can be used to perform each computational step sequentially, reducing the area required by the overall DWT block without impacting performance in this low bandwidth application [6]. This approach requires five cycles to compute results for one input sample pair. Because Q and d in (2) depend on the availability of future samples; the corresponding calculations must be delayed to attain causality, resulting in a latency of three samples

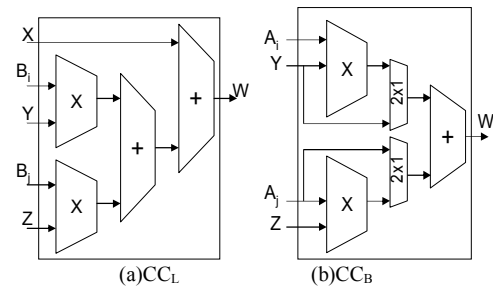


Figure 3. Computation cores for lifting and B-spline architectures.

between input and output.

B. B-spline Implementations

The time dependent equations governing the B-spline factorization using the symmlet4 basis are:

$$\begin{aligned}
 J_0 &= h_0 + (2+4)h_{-1} + h_{-2} & L_0 &= f_0 + (2+4)f_{-1} + f_{-2} \\
 K_0 &= (4)h_{-1} + (4)h_{-2} & M_0 &= (4)f_0 + (4)f_{-1} \\
 N_0 &= J_0 + K_0 & P_0 &= J_0 - K_0 \\
 Q_0 &= L_0 + M_0 & R_0 &= L_0 - M_0 \\
 S_0 &= N_0 + A_0N_{-1} & T_0 &= A_1P_0 + A_2P_{-1} \\
 U_0 &= Q_0 + A_3Q_{-1} & V_0 &= A_4R_0 + A_5R_{-1} \\
 a_0 &= S_0 + T_0 & d_0 &= U_0 + V_0
 \end{aligned} \quad (3)$$

where a and d are the approximation and detail results, A_0 to A_5 are constant coefficients, the variable subscripts represent the time samples, and the terms in parentheses are the shift-and-add multiplications. Because the computations required by (3) vary significantly (e.g., one subtraction, three additions, two multiplications and one addition), they can not be directly mapped to a single computation core. For a hardware minimized implementation that balances hardware utilization and throughput, notice that all equations in (3) can be implemented by $W=A_iY+A_jZ$ or $W=Y\pm Z$ (where A_i and A_j are constant coefficients) in one or two cycles. Fig 3(b) describes a circuit that implements these two expressions, forming the B-spline computation core (CC_B) comparable to the CC_L for the lifting scheme. Using CC_B , the B-spline algorithm can be handled by sequential reuse of two multipliers, an adder and two multiplexers that bypass the multipliers for several filters steps to save power. The shifters needed for binary integer multiplications incur only minor hardware demand and are ignored in subsequent studies.

IV. DWT IMPLEMENTATION

A. Single Channel/Level Implementation

For integer lifting DWT, it has been shown that quantizing data to 10 bits and filter coefficients to 6 bits (including a sign bit) maintains high signal to noise ratio [5]. This quantization will be assumed for both schemes.

Due to the sequential reuse of CC hardware used in order to minimize hardware demand in implantable applications intermediate results from evaluation of (2) and (3) must be stored in *computation core memory* for use in a subsequent filter steps or future CC cycle. A careful analysis shows that, for lifting-based DWT, a total of seven 10-bit registers are required for the computation core memory. For B-spline, 12 10-bit registers are needed to sequentially utilize the CC_B block. A separate *coefficient memory* block is required to store the 5-bit filter coefficients. For lifting, the coefficient memory needs eight registers, while for B-spline only six registers are required. Thus, in both schemes, the overall computational node (CN) necessary to calculate the DWT results consists of one of the computation cores described above, a digital control block, a filter coefficient memory, and a computation core memory.

B. Multi-channel/Level Implementation

Data streams from multiple microelectrodes are preferred for most neuroprosthetic applications, requiring multi-channel data to be compressed simultaneously in real time. In addition, multiple levels of decomposition are preferred to improve the signal fidelity of wavelet transform data. Therefore, implementation constraints for multi-channel, multi-level DWT must be defined and an optimal solution identified.

Recalling that the required sampling rate for neurologically relevant data is quite low, ~ 25 kHz, design constraints can be relaxed by computing multi-channel DWT pseudo-simultaneously, i.e., processing samples from all channels sequentially within the 25kHz sampling period. Analysis of preliminary DWT implementations [6] indicates that there is sufficient bandwidth within a single computation core to process many data channels sequentially. This approach significantly reduces computational hardware demand; however, the intermediate values, or ‘state’, of the current channel needs to be stored in memory so they are available when the CC later processes the next sample from this channel. Similarly, memory is also needed while switching between different levels. The memory block required to hold intermediate values while stepping through different channels or levels is called the *channel/level memory*.

Equations (2) and (3) define which values need to be stored in channel/level memory and made available to process future samples. Lifting requires four 10-bit registers, and B-spline requires eight 10-bit registers for each channel or level. The four additional registers required by B-spline become a key limitation as the number of channels continues to grow in modern neural recording arrays. Calculation of results at each level requires two input samples. For the multilevel case, all levels beyond the first can be computed in the idle state while the DWT unit is loading the first of the sample pair [5].

V. COMPARATIVE ANALYSIS

Table I summarizes the hardware and timing characteristics of both architectures when implemented with the CC modules described above. The multipliers were optimized for 10x6 bits to minimize transistor count and reduce delay compared to a standard square multiplier. The B-spline shifters are listed in Table I but omitted in subsequent calculations because their size and delay is negligible.

The lifting approach results in a highly regular set of equations, and the corresponding CC has high hardware utilization over the five required cycles per sample (per channel). In contrast, the lack of regularity in the B-spline equations forces sequential computations to be stretched out over 18 cycles using an area-minimized CC unit. (Note that with an additional adder in the B-spline CC module, the number of cycles per sample can be reduced to 11. But this is still significantly more than the lifting CC, and it would have

very poor hardware utilization.) The total computational load per sample is the measure of computational resources used to process a single sample, which for the CC_L is $5D_m+10D_a$ and for CC_B is $4D_m+18D_x+18D_a$. As determined by simulations of custom circuits, the delay of the multiplier D_m is three times the delay of the adder D_a and results in a total computational load of $25D_a$ and $30D_a$, for lifting and B-spline, respectively.

TABLE I. HARDWARE COMPARISON OF ARCHITECTURES.

	Lifting CC_L	B-spline CC_B
Multipliers	2	2
Adders	2	1
Multiplexers	0	2
Shifters	0	4
Latency	3	0
Cycles per Sample	5	18
Coeff. Mem. (5bit)	8	6
CC Mem. (10bit)	7	12
No. of Tx for CC	1786	1650
Per Chn/Lvl Mem.	4	8
Critical Path Delay	D_m+2D_a	$D_m+D_x+D_a$

To minimize power consumption, the DWT block is assumed to operate at the minimum frequency necessary to process one sample per channel within the sample period. This results in a different clocking frequency for each of the architectures. For a constant clocking frequency, source voltage and IC fabrication parameters, the power dissipation is directly proportional to the number of transistors active summed over all cycles. The change in clocking frequency linearly scales the power dissipation which is higher for B-spline implementation. Table II lists the number of transistors required for each component of the overall CN. The total transistors in the CN varies with the maximum levels/channels the circuit can process, and Table II includes an example total value for a two channel, one level configuration.

TABLE II. NUMBER OF TRANSISTORS PER BLOCK.

	Lifting	B-spline CC_B
Controller	504	562
Coeff Mem.	240	180
Chn/Lvl Mem.	240	480
CC Mem.	420	720
CC	1786	1652
Total Tx for 2 Chn / 1 Lvl	3190	3594

Based on the data in Table II and empirically derived process constants for a $0.18\mu\text{m}$ CMOS technology, Fig. 4 compares the relative area and power consumption of the B-spline and lifting designs for an increasing number of channels and decomposition levels. The plots show that lifting requires smaller area and consumes less power than B-spline. The CC_B for B-spline requires one less adder than

lifting and thus has fewer transistors. However, in a multi-channel, multi-level implementation the number of transistors for the entire computational node, including memory and controller, increases much more rapidly with B-spline than lifting, offsetting any advantage gained by a smaller CC. For a single-channel, single-level DWT, no channel/level memory is required.

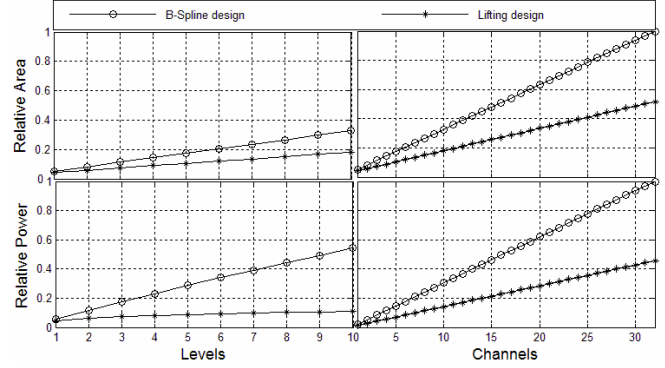


Figure 4. Relative area and power consumption vs. levels and channels.

VI. CONCLUSIONS

Lifting-based and B-spline-based DWT using ‘symmlet4’ wavelet filters were shown to minimize the required area for hardware by sequential evaluation of filter equations. Analysis shows that the B-spline implementation has lower latency and slightly superior performance in single channel/level applications. However, the lifting architecture has lower memory requirements and outperforms the B-spline architecture in all important characteristics in multi-channel, multi-level applications. Because the lifting architecture exhibits better scalability, it is the favored option for implementing data compression within an implanted neuroprosthetic device, and it is increasingly preferred as the number of channels and/or levels rise.

REFERENCES

- [1] K. G. Oweiss, “A systems approach for data compression and latency reduction in cortically controlled brain machine interfaces,” *IEEE Tran. on Biomed. Eng.*, vol. 53, no. 7, pp. 1364-1377, 2006.
- [2] H. Liao, M. K. Mandal, and B. F. Cockburn, “Efficient architectures for 1-D and 2-D lifting-based wavelet transforms,” *IEEE Trans. Signal Process.*, vol. 52, no. 5, pp. 1315-1326, May 2004.
- [3] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, “VLSI architecture for forward discrete wavelet transform based on B-spline factorization,” *J. VLSI Signal Process.*, vol. 40, pp. 343-353, 2005.
- [4] S. Mallat, “A Wavelet Tour of Signal Processing”, New York: Academic, 1998.
- [5] K. Oweiss, A. Mason, K. Thomson, Y. Suhail, and A. Kamboh, “A Scalable Wavelet Transform VLSI Architecture for Real-Time Neural Signal Conditioning in Implantable Multichannel Neuroprosthetic Devices,” submitted to *IEEE Trans. Circuits and Systems*.
- [6] A. Mason, J. Li, K. Thomson, Y. Suhail, and K. Oweiss, “Design optimization of integer lifting DWT circuitry for implantable neuroprosthetics,” *IEEE-EMBS Int. Conf. on Microtechnologies in Medicine and Biology*, pp. 112-115, May 2005.